3 次元 IC 向け Fat Tree ベース Network-on-Chips

松谷宏紀[†]　鯉渕道紘[††]　天野英晴[†]

半導体技術の進歩により Network-on-Chip（NoC）に搭載できるコアの数が増える一方，配線遅延の問題はますます深刻化している．そこで注目を浴びているのがチップの 3 次元化である．3 次元 IC では，チップを垂直方向に積み重ねることで個々のチップサイズを小さくできる．

我々は 3 次元 NoC 向けに Fat Tree ベースのトポロジを検討している．ツリー型トポロジは，ルート付近のリンクの長さが長くなるため配線遅延の点で不利であると考えられてきたが，3 次元化によってこの問題を大幅に改善できることが分かった．本論文では，Fat Tree および Fat H-Tree トポロジを 3 次元 NoC 向けに効率的にレイアウトする方法を提案し，チップ面積，配線量，配線遅延，リピータ数，消費電力について同サイズの 2 次元レイアウトと比較した．その結果，1) 配線量は 25.0% から最大 50.0% 削減された，2) 配線遅延が小さくなり，挿入されるリピータの数が減った，3) これによりフリットの転送エネルギーが最大 59.7% 削減された，4) 3 次元化によってチップ面積が最大 5.0% 増加したが，これはメッシュの 3 次元化に比べて十分低コストである，という点を確かめた．

Fat Tree Based Network-on-Chips for 3-D ICs

Hiroki Matsutani,[†] Michihiro Koibuchi[††]
and Hideharu Amano[†]

The advance of the semiconductor technology allows us to integrate a number of processing cores in a Network-on-Chip (NoC), while the wire delay is increasingly posing a severe problem to the chip design; Therefore, 3-D IC technology that stacks several small-sized ICs in vertical direction becomes an attractive solution to the problem.

For 3-D NoCs, we have been studying Fat Tree based topologies which have been considered to cause a serious wire-delay problem because of long links around their root. In this paper, we propose 3-D layouts of trees in order to resolve their intrinsic disadvantage. The 3-D layouts are compared with original 2-D layouts in terms of chip area, wire length, wire delay, number of repeaters inserted, and energy consumption. The results show that 1) total wire length is reduced by 25.0% to 50.0%; 2) wire delay becomes short and a lot of repeaters which eat much more energy are removed; 3) flit transmission energy is reduced by up to 59.7%; 4) area overhead is at most 5.0%, which is quite smaller than that for 3-D mesh.

1. はじめに

半導体技術の進歩により，単一チップ上にプロセッサやメモリ，I/O など複数の設計モジュールをタイル状に実装できるようになった．このようなタイルアーキテクチャでは，タイル同士の結合に Network-on-Chip（NoC）[1] が用いられ，そのネットワークトポロジはアプリケーションの性能と面積，消費電力を決定付ける一要素となっている．

NoC の規模は年々増大しているが，その一方で，配線遅延や消費電力，それに伴う発熱の増加がますます深刻化している．これまで長い配線にリピータバッファを挿入することで配線遅延の問題を緩和してきたが，遅延の削減効果には限界があるうえに，リピータの挿入は消費電力の増加を引き起こす．したがって，リピータの挿入はこれらの問題の根本的な解決策にはなっていない．

そこで近年注目を浴びているのが，チップの 3 次元化[2)~4)] である．3 次元 IC は複数枚のウェハまたはダイを垂直方向に重ね合わせることで実現される．複数枚のチップを垂直方向に積み重ねることで，個々のチップの寸法を小さく抑えることができ，結果的に配線長および配線遅延を削減できる．例えば，**図 1** の左図は通常の 2 次元 IC を，右図はそれと同じ面積の 3 次元 IC を

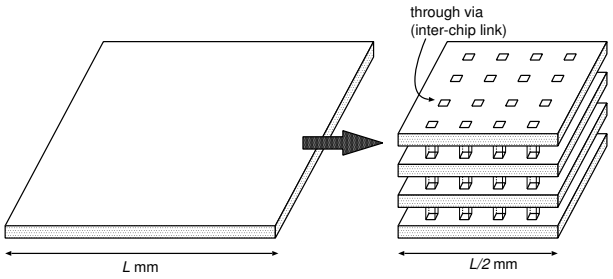

図 1　2 次元 IC（左図）と 3 次元 IC（右図）．

示している．この 3 次元 IC では 4 枚のチップを積層する代わりに，個々のチップの面積は 1/4，チップの端と端を結ぶ配線長は 1/2 に削減される．配線遅延は配線長の 2 乗に比例するため，配線遅延は最大で 1/4 まで削減され，それに伴い必要なリピータバッファの数も減る．

3 次元 IC には，ダイ同士をワイヤボンディングで結合するもの，マイクロバンプで結合するもの，チップ間無線通信を用いるもの，また，ウェハ同士を貫通ビアで結合するものなどがある[2)]．本論文ではチップ間リンクを最も高密度に実装できると期待されている貫通ビアを想定して議論を進める．文献 2) によると，この貫通ビアによる方法では，チップ間の間隔は 10μm 程度まで抑えることができるため，チップ間リンクは水平方向のリンク長に比べて無視できるほど短くなる．一方，貫通ビアのサイズは 1μm 角〜10μm 角[2)~4)] である．これは配線ピッチよりはるかに大き

† 慶應義塾大学大学院 理工学研究科
Graduate School of Science and Technology, Keio University
†† 国立情報学研究所
National Institute of Informatics

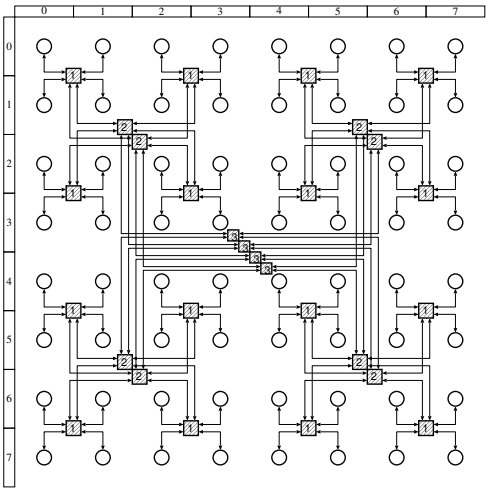

図 2 $(2, 4, 1)$ Fat Tree.

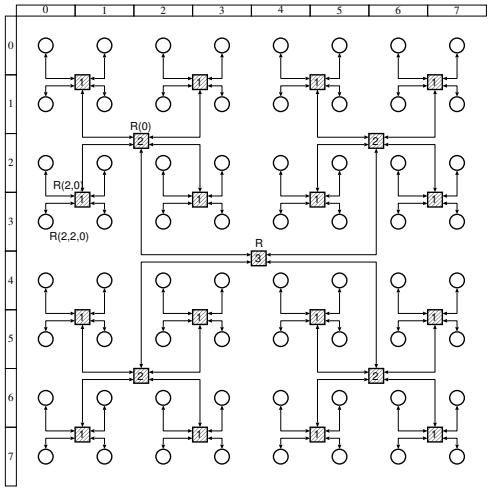

図 3 Fat H-Tree（red tree）.

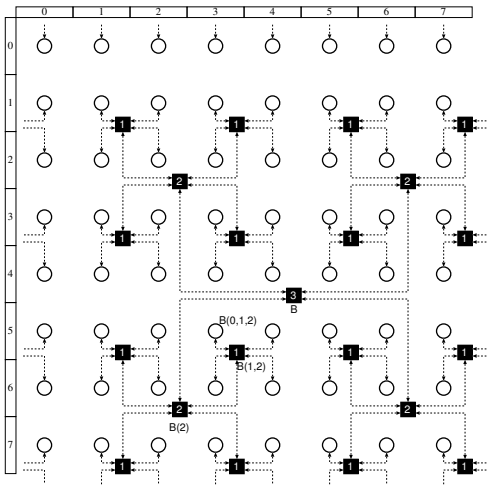

図 4 Fat H-Tree（black tree）.

いので，チップ間リンクの数によってはその面積が問題になる．

3 次元 IC を想定した NoC の研究はまだ数が少ないが，3-D Mesh をベースにしたものがいくつか報告されている[5]．3-D Mesh や 3-D Torus は通常の 2 次元トポロジに比べ高いスループットが得られるが，次のような欠点が指摘されている[3]．

- 3 次元 IC では垂直方向のリンクは数十 μm 程度と非常に短いため，3-D Mesh のように垂直方向にオンチップルータを積み重ね，データをチップ毎にラッチする必要はない．
- 3-D Mesh では 2-D Mesh と比べ，オンチップルータの次数が 4 から 6 に増え，ルータのチャネルバッファおよびクロスバスイッチの面積が増加する．

そこで，より低コストな NoC として，我々は 3 次元 IC 向けに Fat Tree ベースのトポロジを検討している．これらは，3-D Mesh や 3-D Torus に比べて計算コアを含めたルータのポート数が少なく，ルータを簡素化できる利点を持つ．一方で，これまでツリー型トポロジは，ルート付近のリンクの長さが長くなるため配線遅延の点で不利であると考えられてきたが，3 次元化によってこの弱点を克服できることが分かった．本論文では，Fat Tree および Fat H-Tree トポロジを 3 次元 NoC 向けに効率的にレイアウトする方法を提案し，3 次元化によって配線量，配線遅延，リピータ数，消費電力がどれだけ削減できたかを示す．

本論文の構成は次のとおりである．まず 2 章で既存の NoC のトポロジについて述べ，3 章と 4 章で Fat Tree および Fat H-Tree の 3 次元 IC 向けレイアウトを提案する．5 章で評価を行い，6 章で本論文をまとめる．

2. NoC のトポロジ

2.1 Mesh と Torus

通常の 2 次元 IC においては 2-D Mesh や 2-D Torus などのグリッド型のトポロジが広く用いられている[1]．3 次元 IC においても 3-D Mesh をベースとした NoC の研究例[5] があるものの，1 章で指摘したとおり，2 次元 IC 向けレイアウト（2-D Mesh）に比べハードウェア量の増加が著しい．

2.2 Fat Tree

Fat Tree はツリーを多重化することで，ツリーのルート付近のトラフィックの集中を防ぐ（**図 2**）．図中の丸は計算コアを，四角はルータを表し，四角の中の数字はルータのランク数である．

Fat Tree には様々な形態があるため，ここでは Fat Tree を（上向きリンク数 p，下向きリンク数 q，コアの上向きリンク数 c）の組みで表記する．図 2 では，各ルータは 2 本の上位リンク，4 本の下位リンク，各コアは 1 本の上位リンクを持つため，以後この Fat Tree を $(2, 4, 1)$ Fat Tree と表記する．

Fat Tree では最上位リンクの配線長が最も長くなる．このため，リンクの配線長が均一なグリッド型トポロジに比べて，配線遅延やリピータ数，消費電力の点で不利となる．

2.3 Fat H-Tree

Fat H-Tree[6),7)] は red tree と black tree と呼ばれる 2 つの H-Tree を組み合わせたトポロジであり，トーラス構造を内包している．各計算コアは 2 つのポートを持ち，1 つを red tree との接続に，もう片方を black tree との接続に用いる．Fat H-Tree は Fat Tree よりコスト性能比で優れることが確認されている[7]．

a) Red Tree の定義

図 3 に red tree の例を示す．丸で描かれた各計算コアには 2 次元座標 (x_{2D}, y_{2D}) がそれぞれ割り当てられる．以降，計算コアを rank 0 ルータ呼ぶ．まず，rank 0 ルータに対し次の式で求まる **red tree 座標** $R(r_0, r_1, ..., r_{n-1})$ を割り当てる．

$$r_i = ((x_{2D}/2^i) \bmod 2) + 2 \times ((y_{2D}/2^i) \bmod 2) \qquad (1)$$

red tree 座標 $R(r_0, r_1, ..., r_{n-1})$ となる rank 0 ルータの上位ルータを rank 1 ルータと呼び，$R(r_1, ..., r_{n-1})$ の red tree 座標を割り当てる．同様の方法で rank 2 ルータから rank n ルータにも red tree 座標を割り振る．ただし，最上位ランク（rank n）ルータの red tree 座標は R とする．例として 図 3 に R，$R(0)$，$R(2, 0)$，$R(2, 2, 0)$ の座標を示す．

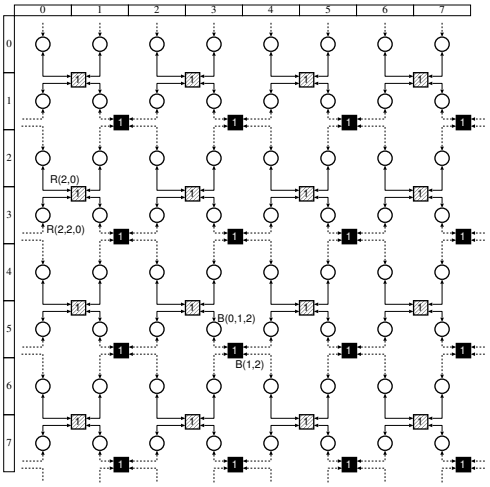

図 5 Fat H-Tree（rank 2 以上のルータは省略）.

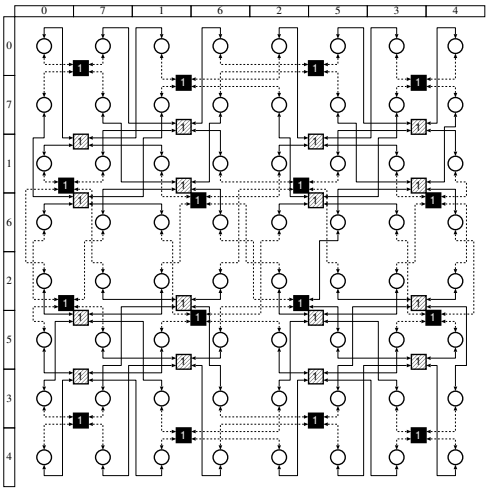

図 6 Folded Fat H-Tree（rank 2 以上のルータは省略）.

b) Black Tree の定義

図 4 に black tree の例を示す．black tree は red tree の座標を右斜め下方向に 1 つずらしたものである．このため rank 0 ルータの **black tree 座標** $B(b_0, b_1, ..., b_{n-1})$ は次の式で求まる．

$$b_i = ((((x_{2D} - 1) \bmod 2^n)/2^i) \bmod 2) + 2 \times ((((y_{2D} - 1) \bmod 2^n)/2^i) \bmod 2) \quad (2)$$

black tree においても rank 1 ルータから rank n ルータに black tree 座標を割り当てていく．最上位ランク（rank n）ルータの black tree 座標は B とする．

c) Fat H-Tree の定義

各計算コア（rank 0 ルータ）に対し，red tree および black tree を接続したものを **Fat H-Tree** と呼ぶ．

Fat H-Tree においては rank 0 ルータと red tree および black tree の rank 1 ルータによってトーラス構造が形成される．Fat H-Tree が内包するトーラス構造を **図 5** に示す．

d) 2 次元レイアウト

Fat H-Tree は，トーラス同様，畳み込むことで 2 次元チップ上に容易にレイアウトできる．**図 6** に Fat H-Tree の 2 次元レイアウトを示す．図 5 と 図 6 は等価なトポロジである．

最上位リンクを除き，Fat H-Tree の各リンクは畳み込みによって配線長が約 2 倍に延びるが，最上位リンクの配線長は大幅に短縮される．そのため，最上位の次のランクのリンク（第 2 位リンク）が Fat H-Tree の最長配線となる[7]．

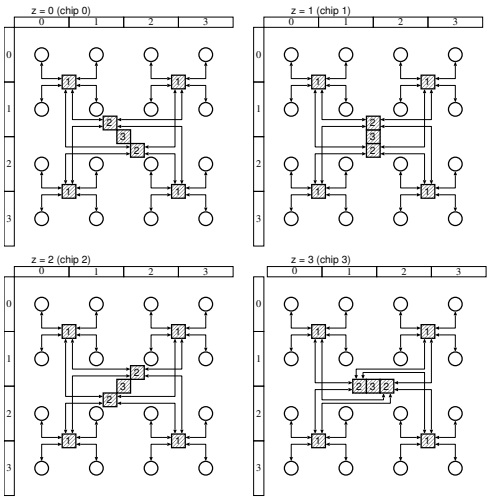

図 7 3-D $(2, 4, 1)$ Fat Tree.

3. Fat Tree の 3 次元 IC 向けレイアウト

Fat Tree の弱点は，最上位リンクの配線長が長くなることによる配線遅延の増加，リピータ数の増加，およびそれに伴う消費電力の増加である．これらの問題を同時に解決するために，図 2 で示した Fat Tree を半分の寸法から成る 4 枚のチップに分割し，これらを 3 次元的に結合する．こうすることで，すべての最上位リンクは距離数十 μm 程度のチップ間リンクに置き換えられ，配線長の問題は解決される．

3.1 Fat Tree の 4 分割

Fat Tree の 4 分割手順は次のとおりである．

(1) （チップの分割）コア数を $2^n \times 2^n$ 個とするとき，各コアの 2 次元座標 (x_{2D}, y_{2D}) を，次のような 3 次元座標 (x_{3D}, y_{3D}, z_{3D}) に変換する．

$$x_{3D} = x_{2D} \bmod 2^{n-1}$$
$$y_{3D} = y_{2D} \bmod 2^{n-1}$$
$$z_{3D} = 2 \times \lfloor y_{2D}/2^{n-1} \rfloor + \lfloor x_{2D}/2^{n-1} \rfloor$$

例えば，図 2 の 64 コア Fat Tree は **図 7** のように 4 枚の 16 コアチップに分割される．

(2) （ルータの配置）既存の 2 次元レイアウト（図 2）をもとに，ルータを各チップに均等に振り分ける．図 7 の例では，各チップは 4 個の rank 1 ルータ，2 個の rank 2 ルータ，1 個の rank 3 ルータを持つ．

(3) （チップ間配線）他階層に配置されたルータ同士をつなぐ最上位リンクは，マイクロバンプ，無線，貫通ビアなどのチップ間通信技術[2] を用いて結合される．

(4) （ルータの再配置）垂直方向のチップ間リンク同士が重ならないように，ルータの位置を調整する．図 7 のように，チップの中心に rank 3 ルータを置き，その周囲に他の階層と重ならないように rank 2 ルータを密接に配置することで配線長を抑えている．

これにより最上位リンクはチップ間リンクに置き換えられる．結果的に，最上位リンクの 1/2 の長さの第 2 位リンクが最長配線となるため，4 分割によって最長配線長は 1/2 に短縮される．

この方法は，$(2, 4, 2)$ など他の構成の Fat Tree に対しても適用できる．4 分割以外の分割方法については次節で示す．

3.2 それ以外の分割

i を正の整数とするとき，チップの 2^i 分割はチップの 4 分割と 2 分割を組み合わせることで実現できる．チップの 2 分割の場

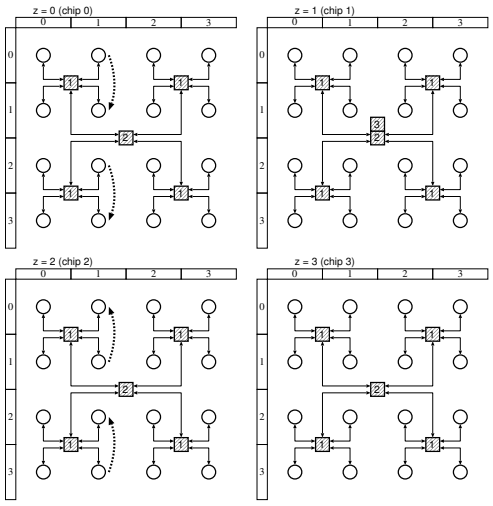

図 8 3-D Fat H-Tree（red tree）.

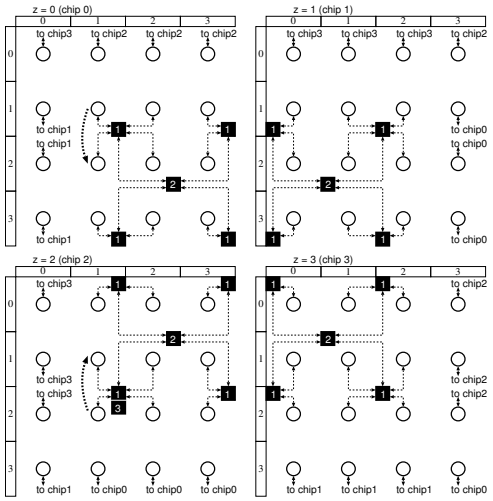

図 9 3-D Fat H-Tree（black tree）.

合，2 次元のコア座標から 3 次元座標への変換は次のようになる．

$$x_{3D} = x_{2D}$$
$$y_{3D} = y_{2D} \bmod 2^{n-1}$$
$$z_{3D} = \lfloor y_{2D}/2^{n-1} \rfloor$$

これ以外の点については 4 分割の場合と同じである．

チップ間リンクの距離を無視するとき，最上位リンクの長さは 2 分割によって 3/4 に短縮される．

4. Fat H-Tree の 3 次元 IC 向けレイアウト

Fat H-Tree の 2 次元レイアウトでは，畳み込みによって第 2 位リンクが最長配線となる．この最長配線は Fat Tree の最長配線の長さと同じであり，配線遅延，リピータ，および消費電力の増加を引き起こす．本章では，Fat Tree 同様，これらの問題を 3 次元化によって解決する．

4.1 Fat H-Tree の 4 分割

Fat H-Tree は 2 個の H-Tree の他に 2-D Torus 構造を内包するため，3 次元空間においてもこの 2-D Torus 構造が保存されていなければならない．これは Fat H-Tree の 2 次元コア座標から 3 次元座標に変換するときに，転置処理を加えることで実現する．

Fat H-Tree の 4 分割手順は次のとおりである．

(1) （チップの分割）2 次元 Fat H-Tree のコア座標 (x_{2D}, y_{2D}) を，次のような 3 次元座標 (x_{3D}, y_{3D}, z_{3D}) に変換する．

$$x_{3D} = \begin{cases} x_{2D} \bmod 2^{n-1} & x_{2D} < 2^{n-1} \\ 2^{n-1} - (x_{2D} \bmod 2^{n-1}) & x_{2D} \geq 2^{n-1} \end{cases}$$
$$y_{3D} = \begin{cases} y_{2D} \bmod 2^{n-1} & y_{2D} < 2^{n-1} \\ 2^{n-1} - (y_{2D} \bmod 2^{n-1}) & y_{2D} \geq 2^{n-1} \end{cases}$$
$$z_{3D} = 2 \times \lfloor y_{2D}/2^{n-1} \rfloor + \lfloor x_{2D}/2^{n-1} \rfloor$$

この処理によって，元のチップはチップの中心を軸に水平方向，および，垂直方向に折り畳まれ 4 枚のチップに分割される．例えば，図 3 の red tree は **図 8** のように，また，図 4 の black tree は**図 9** のようにそれぞれ分割される．

(2) （ルータの配置）ルータを各チップに均等に振り分ける．この例では，red tree（図 8）および black tree（図 9）ともに，各チップに 4 個の rank 1 ルータ，1 個の rank 2 ルータを持つ．一方，rank 3 ルータは，red tree は chip 1 に，black tree は chip 2 にそれぞれ 1 個だけ持つ．

(3) （チップ間配線）他階層に配置されたルータ同士をつなぐ．black tree（図 9）では最上位ルータに加え，rank 1 ルータも異なる階層のコア同士をつないでいる．

(4) （ルータの再配置）必要に応じて行う．

図 8 および 図 9 を組み合わせたものが Fat H-Tree の 3 次元レイアウトであり，分割後もトーラス構造が保存されている．図中の点線矢印の例では，コア $(1,0,0)$ の $y+$ 方向 2 ホップ先はコア $(1,1,0)$ であり，その次はコア $(1,2,0)$，コア $(1,3,0)$ と移動した後，chip 2 に移り，コア $(1,3,2)$，コア $(1,2,2)$，コア $(1,1,2)$，コア $(1,0,2)$ を経て，再び chip 0 のコア $(1,0,0)$ に戻る．

4.2 それ以外の分割

チップの 2 分割の場合，2 次元のコア座標から 3 次元座標への変換は次のようになる．

$$x_{3D} = x_{2D}$$
$$y_{3D} = \begin{cases} y_{2D} \bmod 2^{n-1} & y_{2D} < 2^{n-1} \\ 2^{n-1} - (y_{2D} \bmod 2^{n-1}) & y_{2D} \geq 2^{n-1} \end{cases}$$
$$z_{3D} = \lfloor y_{2D}/2^{n-1} \rfloor$$

この場合，x 方向のリングは単一チップ内で完結するため，x 方向に対してのみ 図 6 のような畳み込みを行う．これ以外の点については 4 分割の場合と同じである．

チップ間リンクの距離を無視するとき，最上位リンクの長さは 2 分割によって 3/4 に短縮される．

5. 評　　価

本論文で提案した Fat Tree および Fat H-Tree の 3 次元レイアウト（**3-D Fat Tree** および **3-D Fat H-Tree**）をチップ面積，配線量，配線遅延，リピータ数，消費電力について評価する．これらの 3 次元レイアウトは 2^i 枚のチップ上に実現できるが，紙面の都合によりここでは 4 枚の場合の評価結果のみを示す．

5.1 ルータの次数と個数

本節では，3-D Fat Tree および 3-D Fat H-Tree のチップ面積を評価する前段階として，これらのレイアウトに必要となるルータの個数を見積もった（**表 1**）．Fat H-Tree をはじめとするツリー型トポロジでは，Mesh や Torus に比べてルータの数が少ない．加えて，個々のルータの最大ポート数は H-Tree と Fat H-Tree で 5 個，Fat Tree で 6 個であるのに対し，3-D Mesh と 3-D Torus では 7 個となり，ルータ 1 個当たりの面積でもツリー型トポロジのほうが有利である．ただし，(2,4,2) Fat Tree と Fat H-Tree では各コアは 2 本のリンクを持つため，他のトポロジと比べネットワークインターフェイス（NI）の面積が増加する．NI を含めたチップ面積は次節で示す．

表 1 ルータの個数.

	N-core	16-core	64-core	256-core
H-Tree	$(4^n-1)/3$	5	21	85
Fat Tree (2,4,1)	$(4^n-2^n)/2$	6	28	120
Fat Tree (2,4,2)	4^n-2^n	12	56	240
Fat H-Tree	$2(4^n-1)/3$	10	42	170
Mesh	N	16	64	256
Torus	N	16	64	256

5.2 チップ面積

本節では，3-D Fat Tree および 3-D Fat H-Tree における，オンチップルータと NI および貫通ビアの面積を評価する．各トポロジの面積は次の手順で見積もった．1) Verilog-HDL で記述されたルータ回路と NI 回路を組み合わせることで対象トポロジを構築，2) これを 0.18μm スタンダードセルライブラリを用いて Synopsys 社の Design Compiler で合成することで NoC の面積を抽出，3) 各トポロジに含まれるチップ間リンクの数から貫通ビアの面積を計算し，NoC の面積に足し合わせる．

本評価では文献 7) の 32-bit wormhole ルータ回路を用いた．このルータ回路は 4 段のパイプラインステージから構成され，各パイプラインステージごとに 1-flit 分のバッファを持つ．NI 回路として 2-flit 分の FIFO を入力側と出力側にそれぞれ持たせた．Fat H-Tree と (2,4,2) Fat Tree では 2 ポートの NI が必要になる．とくに Fat H-Tree では，片方のポートからもう片方のポートへパケットを転送する機能が必要であり，このために NI 内に小さなマルチプレクサが実装されている．

貫通ビアのサイズはチップを重ね合わせるときのアライメント精度によって制限され[4]，一般的には 1μm 角〜10μm 角[2)~4)] となる．そこで，本評価では，1-bit の単方向チップ間リンクが 1 枚のチップを貫通する度に，貫通ビアの面積として 100μm^2 を加えていった．以上の方法で求めた 3 次元レイアウトの面積を **表 2** に示す．表中のカッコ内の数字は 2 次元レイアウトからの面積の増分である．つまり，この数字はツリー型トポロジでは貫通ビアのみの面積を表し，Mesh や Torus では 3 次元化に伴うルータ面積の増分と貫通ビアの面積を表す．

表 2 より，ツリー型トポロジでは，3 次元化のための貫通ビアによって総面積が最大 5.0%増加している．一方，Mesh と Torus では 3 次元化によってルータのポート数が増えるため，面積が Mesh で最大 25.1%，Torus で最大 47.3% 増加しており，3 次元化のオーバーヘッドが非常に大きいことが分かる．

表 2 チップ面積（カッコ内は 2 次元レイアウトからの増分）[mm^2].

	16-core		64-core	
3-D H-Tree	1.73	(0.03)	7.20	(0.03)
3-D Fat Tree (2,4,1)	2.17	(0.05)	10.16	(0.13)
3-D Fat Tree (2,4,2)	4.37	(0.13)	20.32	(0.26)
3-D Fat H-Tree	3.78	(0.18)	15.46	(0.33)
3-D Mesh	3.88	(0.46)	19.01	(3.81)
3-D Torus	6.16	(1.98)	24.62	(7.90)

5.3 総配線長

本節では，隣接コア間距離を 1-unit としたときの配線量を評価する．まず，通常の 2 次元トポロジの総配線長を計算し，次にそれを 3 次元化したときの総配線長を求めることで，3 次元化による配線長の削減率を示す．

5.3.1 2 次元レイアウト

ここではランク数 n の H-Tree の配線長 $L_{2D,ht}$ を次式で表す．

$$L_{2D,ht} = \sum_{i=1}^{n} l_{ht}^{i} \cdot r_{ht}^{i} \tag{3}$$

ただし，l_{ht}^{i} を H-Tree における rank i ルータからその 4 個の下位ルータへの総リンク長，r_{ht}^{i} を rank i ルータの数とする．計算コア数を $N = 2^n \times 2^n$ とすると，$l_{ht}^{i} = 2^{i+1}$，$r_{ht}^{i} = N/4^i$ となる．したがって，式 3 は次式のように変形できる．

$$L_{2D,ht} = \sum_{i=1}^{n} 2^{i+1} \cdot \frac{N}{4^i} = 2N\left(\frac{2^n-1}{2^n}\right) \tag{4}$$

同様の方法を用いることで，ランク数 n の Fat H-Tree の総配線長 $L_{2D,fht}$ を求める式は次のように導くことができる[7]．

$$L_{2D,fht} = 8 + 8N\left(\frac{2^{n-1}-1}{2^{n-1}}\right) \tag{5}$$

2 次元レイアウトの配線量を **表 3** にまとめる．この見積りでは Mesh/Torus のコア-ルータ間リンクの長さを無視しているものの，Fat H-Tree や (2,4,2) Fat Tree などツリー型トポロジの総配線長が圧倒的に長く，これがツリーの欠点になっている．以降，これが 3 次元化によってどれだけ削減できるかを示す．

表 3 2 次元レイアウトにおける総配線長（隣接コア間距離を 1-unit とする）.

	N-core	16-core	64-core	256-core
2-D H-Tree	式 4	24	112	480
2-D Fat Tree (2,4,1)	nN	32	192	1,024
2-D Fat Tree (2,4,2)	$2nN$	64	384	2,048
2-D Fat H-Tree	式 5	72	392	1,800
2-D Mesh	$2(N-2^n)$	24	112	480
2-D Torus	$4(N-2^n)$	48	224	960

5.3.2 3 次元レイアウト

3 次元レイアウトではチップ間の距離は数十 μm 程度と，水平方向のリンク長に比べて圧倒的に短い．そのため，ここではチップ間リンクの長さは考慮しないものとする．

まず，ランク数 n の H-Tree を 3 次元化したときの総配線長 $L_{3D,ht}$ を求める．チップの枚数は 4 枚なので，すべての最上位リンクが長さ 0 のチップ間リンクに置き換えられることになる．したがって，式 4 は 3 次元化によって次式のように変化する．

$$L_{3D,ht} = \sum_{i=1}^{n-1} 2^{i+1} \cdot \frac{N}{4^i} = 2N\left(\frac{2^{n-1}-1}{2^{n-1}}\right) \tag{6}$$

Fat H-Tree の場合は red tree と black tree に分けて考える．red tree は H-Tree の 3 次元レイアウトと等価である（図 8）．一方，black tree はこれに加え，最上位リンクの長さとしてそれぞれ 2-unit 必要となる（図 9）．よって総配線長は次のとおり．

$$L_{3D,fht} = 8 + 4N\left(\frac{2^{n-1}-1}{2^{n-1}}\right) \tag{7}$$

3 次元レイアウトの総配線量を **表 4** にまとめる．Fat Tree および Fat H-Tree では，3 次元化によって総配線長が 25.0% から最大 50.0% 削減できた．その結果，総配線長は，(2,4,2) Fat Tree を除き，3-D Torus と同程度かそれ以下に削減された．

表 4 3 次元レイアウトにおける総配線長（隣接コア間距離を 1-unit とする）.

	N-core	16-core	64-core	256-core
3-D H-Tree	式 6	16	96	448
3-D Fat Tree (2,4,1)	$(n-1)N$	16	128	768
3-D Fat Tree (2,4,2)	$2(n-1)N$	32	256	1,536
3-D Fat H-Tree	式 7	40	200	904
3-D Mesh	$2(N-2^{n+1})$	16	96	448
3-D Torus	$4(N-2^{n+1})$	32	192	896

5.4 消費電力

1-flit のデータを送信元から宛先ノードに転送するとき，これに要す平均転送エネルギーは次式で計算できる．

$$E_{flit} = wH_{ave}(E_{sw} + E_{link}) \quad (8)$$

ただし，w を 1-flit のビット数，H_{ave} を平均ホップ数，E_{sw} をルータが 1-bit のデータ転送に消費するエネルギー，E_{link} をリンクが 1-bit のデータ転送に消費するエネルギーとする．

5.2 節で合成したルータを，250MHz での動作を仮定してゲートレベルでシミュレーションしたところ，E_{sw} は 1.13pJ となった．一方，E_{link} は次式で計算できる．

$$E_{link} = dV^2C_{wire}/2 \quad (9)$$

ただし，d を 1-hop 当たりの平均距離（mm），V を動作電圧，C_{wire} を配線容量とする．本評価では V を 1.8V とし，C_{wire} は 0.18μm プロセスを仮定するとき 414fF/mm となった[8]．

チップサイズは 12mm 角とし，16 コア，64 コア，256 コアの場合について，上記のパラメータをもとに E_{flit} を計算した．

5.4.1 2 次元レイアウト

リピータバッファの挿入を考慮しないとき，2 次元レイアウトにおける 1-flit 当たりの転送エネルギー E_{flit} は **表 5** のようになった．Fat H-Tree は畳み込みによって 1-hop 当たりの平均距離 d が延びたが，平均ホップ数はこの中で最も小さく，消費電力においても有利となった．H-Tree や Fat Tree はこの次に平均ホップ数が小さいが，最上位リンクの利用率が高いため転送エネルギーは Fat H-Tree よりも大きくなった．2-D Torus は畳み込みによって d が延びるため 16 コアでは 2-D Mesh より転送エネルギーが多い．一方，256 コアでは E_{sw} のウェイトが高くなり，平均ホップ数の小さい Torus は Mesh より有利になっている．

表 5 2 次元レイアウトにおける転送エネルギー E_{flit} [pJ]（リピータ無し）．

	16-core	64-core	256-core
2-D H-Tree	464.6	579.2	676.8
2-D Fat Tree (2,4,*)	464.6	579.2	676.8
2-D Fat H-Tree	424.4	530.6	636.3
2-D Mesh	468.9	501.0	661.7
2-D Torus	483.9	512.3	637.0

チップサイズが 12mm 角のとき，ツリー型トポロジおける最長配線長はコア数によらず常に 6mm である．このときの配線遅延は 715psec となり，プロセスの微細化に伴いさらに深刻化すると考えられる．そこで，文献 8) で示される計算式にもとづきリピータを挿入する．その結果，配線遅延を 441psec まで抑えることができたが，新たにリピータバッファの消費電力を考慮する必要が生じた．ここでは 5mm 以上のリンクにリピータバッファを挿入するものとして E_{flit} を計算した☆．結果を **表 6** に示す．

ツリー型トポロジの最長リンクには平均 4.9 個のリピータが挿入されたため，リピータを用いない場合（表 5）と比べ，転送エネルギーが大幅に増加した．また，16 コアの 2-D Torus においては畳み込みにより一部のルータ-ルータ間リンクが 5mm を越えたため，ツリー型トポロジ同様 E_{flit} が増加した．

表 6 2 次元レイアウトにおける転送エネルギー E_{flit} [pJ]（リピータ有り）．

	16-core	64-core	256-core
2-D H-Tree	642.8	748.8	844.5
2-D Fat Tree (2,4,*)	642.8	748.8	844.5
2-D Fat H-Tree	691.7	641.2	786.0
2-D Mesh	468.9	501.0	661.7
2-D Torus	662.1	512.3	637.0

☆ 文献 8) の図 10 より，4mm 以下の配線にリピータを挿入しても配線遅延の削減効果が小さいことが分かる．そこで消費電力の増加を少しでも抑えるため，リピータの挿入は 5mm 以上のリンクに制限した．

5.4.2 3 次元レイアウト

最後に，3 次元レイアウトにおける転送エネルギーを計算する．文献 2) によると貫通ビアの容量は 4.34fF である．これは本評価で仮定している配線 10.5μm 分の容量に相当し，水平方向の配線長に比べて十分小さい．そのため，チップ間リンクで消費されるエネルギーは考慮しないものとする．

結果を **表 7** に示す．3 次元レイアウトでは，どのトポロジにおいても 5mm を越えるリンクが発生しなかったためリピータは不要となった．Fat Tree の転送エネルギーは 表 5 と比べて最大 44.3%，表 6 と比べて最大 59.7% 削減された．同様に Fat H-Tree は 表 5 と比べて最大 32.8%，表 6 と比べて最大 53.5% 削減された．3 次元化により，配線長およびリピータを減らすことができ，結果的に転送エネルギーを大幅に減らすことができた．

表 7 3 次元レイアウトにおける転送エネルギー E_{flit} [pJ]（リピータ不要）．

	16-core	64-core	256-core
3-D H-Tree	258.8	383.1	483.1
3-D Fat Tree (2,4,*)	258.8	383.1	483.1
3-D Fat H-Tree	321.5	356.6	441.0
3-D Mesh	300.6	334.6	418.0
3-D Torus	282.4	327.0	398.4

6. ま と め

これまでツリー型トポロジは，ルート付近のリンクの長さが長くなるため配線遅延の点で不利であると考えられてきた．この問題を解決するために本論文で提案した Fat Tree および Fat H-Tree の 3 次元化レイアウトは，同サイズの 2 次元レイアウトに比べ，1) 配線量は 25.0% から最大 50.0% 削減された，2) 配線遅延が小さくなり，挿入されるリピータの数が減った，3) これによりフリットの転送エネルギーが最大 59.7% 削減された，4) 3 次元化によってチップ面積が最大 5.0% 増加したが，これはメッシュの 3 次元化に比べて十分低コストである．

すでに多くの 3 次元 IC が実用化されているが，3 次元化の問題点として，歩留まりの低下や放熱の難しさなどが指摘されている[2]．3 次元 IC の普及には，これらの問題の解決が必要であると考えられる．

参 考 文 献

1) Dally, W. J. and Towles, B.: Route Packets, Not Wires: On-Chip Interconnection Networks, *Proceedings of the Design Automation Conference*, pp. 684–689 (2001).
2) Davis, W. R. and et. al.: Demystifying 3D ICs: The Pros and Cons of Going Vertical, *IEEE Design and Test of Computers*, Vol. 22, No. 6, pp. 498–510 (2005).
3) Li, F. and et. al.: Design and Management of 3D Chip Multiprocessors Using Network-in-Memory, *Proceedings of the International Symposium on Computer Architecture*, pp.130–141 (2006).
4) Das, S. and et. al.: Technology, Performance, and Computer-Aided Design of Three-Dimensional Integrated Circuits, *Proceedings of the International Symposium on Physical Design*, pp. 108–115 (2004).
5) Addo-Quaye, C.: Thermal-Aware Mapping and Placement for 3-D NoC Designs, *Proceedings of the International System-on-Chip Conference*, pp. 25–28 (2005).
6) 山田裕, 天野英晴, 鯉渕道紘, 上樂明也, 安生健一朗: リコンフィギャラブルプロセッサアレイ向けチップ内接続網: Fat H-Tree, 電子情報通信学会論文誌 D, Vol. J89-D, No. 9, pp. 1923–1934 (2006).
7) Matsutani, H., Koibuchi, M. and Amano, H.: Performance, Cost, and Energy Evaluation of Fat H-Tree: A Cost-Efficient Tree-Based On-Chip Network, *Proceedings of the International Parallel and Distributed Processing Symposium* (2007). (to be appeared).
8) Ho, R., Mai, K. W. and Horowitz, M. A.: The Future of Wires, *Proceedings of the IEEE*, Vol. 89, No. 4, pp. 490–504 (2001).